UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/731,281	12/09/2003	Jianbing Huang	05-03-014	4973

45113 7590 08/14/2008

DOCKET CLERK
PO BOX 800889
DALLAS, TX 75380

EXAMINER	
NGUYEN, KIMBINH T	
ART UNIT	PAPER NUMBER
2628	
MAIL DATE	DELIVERY MODE
08/14/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte JIANBING HUANG and MICHAEL B. CARTER

Appeal 2008-1410
Application 10/731,281
Technology Center 2600

Decided: August 14, 2008

Before KENNETH W. HAIRSTON, JOHN C. MARTIN, and LEE E. BARRETT, *Administrative Patent Judges.*

MARTIN, *Administrative Patent Judge.*

DECISION ON APPEAL

STATEMENT OF THE CASE

This is an appeal under 35 U.S.C. § 134(a) from the Examiner's rejection of claims 1-30 under § 103(a).

We have jurisdiction under 35 U.S.C. § 6(b).

We AFFIRM-IN-PART.

A. Appellants' disclosed invention

The invention relates to graphics processing and more particularly to a system, method, and computer program product that accepts raw polygon geometry and view parameters from a visualization API, sorts the polygons in back-to-front order, and then supplies the sorted triangles to a graphics API such as OpenGL (Specification 4 [0008][1]). Although triangles are specifically discussed, any general planar shapes can be used (*id.* at 8 [0025]).[2]

Prior-art methods for sorting triangles can be classified as two types: image-space based and object-space based (*id.* at 2 [0004]). In the image-space based methods, which require special graphics hardware, the polygons are rendered layer by layer along the depth in back-to-front order (*id.*). The rendering of each layer requires a separate rendering pass, and the number of passes required is the maximum depth of the scene (*id.*). Binary space partitioning (BSP), which is an object-space based method, sorts all of the polygons with respect to the current view direction, with the sorted polygons

---

[1] Reference in this opinion to entire paragraphs of the Specification are to the page numbers and the paragraph numbers, which are in brackets. A reference to only part of a paragraph is to the page and line numbers (e.g., "11:3-5").

[2] While the Specification more particularly discusses "transparent polygons" (*e.g.*, *id.* at 2:8) and "transparent triangles" (*e.g.*, *id.* at 14:2), transparency is not required by the claims and thus not discussed below.

then being supplied to the graphics pipeline in back-to-front order for rendering (*id.*).

The Specification describes the prior-art technique of forming a BSP tree as follows:

> [E]ach node represents a set of elements located in a 3-dimensional spatial region. Starting with the root node containing all the elements, a partition plane is used to divide the whole Euclidean space into two non-overlapping regions. Those elements cut by the partition plane will be split into two or more elements. For the elements in each region a child node is created which becomes a new leaf node of the BSP tree. This partitioning continues recursively for each leaf node until some pre-set conditions are met.

Specification 14:27-15:5.[3]

Appellants' Figure 4A is reproduced below.

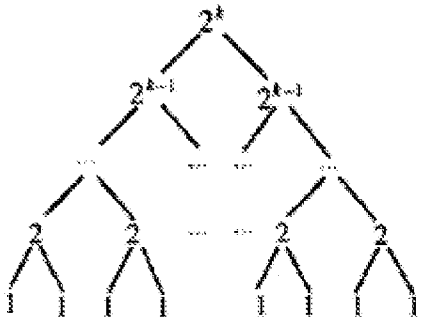


Figure 4A

[3] The basics of BSP construction are also described in Henry Fuchs, et al., *Near Real-Time Shaded Display of Rigid Objects*, Computer Graphics, Vol. 17 (July 1983), pp. 65-72, cited in Appellants' Information Disclosure Statement filed August 5, 2005.

Figure 4A shows an "extremely balanced" tree partition (*id.* at 18:17-18). In this example, the total number of triangles is $N = 2^k$, no triangle splitting occurs during BSP construction (*id.* at 18, 20-22), and "each leaf node contains exactly one triangle" (*id.* at 18:22-23).

Appellants' invention integrates the BSP sort method with the depth sort method. *Id.* at 11:5-11. Specifically,

> in at least some embodiments, a BSP tree is constructed with multiple-triangle leaf nodes (as opposed to a BSP tree with single-triangle leaf nodes as in conventional practices), and the triangles are sorted by traversing the BSP tree and depth-sorting triangles on each leaf node while it is being traversed.

Id. at 11:12-17. More particularly,

> [b]y specifying the maximum number of triangles β allowed on a single leaf node, the behavior of the disclosed algorithm can be readily tuned between these two methods to achieve the desired tradeoff between rendering speed and memory consumption. For example, the disclosed algorithm degenerates into [a] traditional BSP sort algorithm if β is chosen to be 1, and degenerates into [a] traditional depth sort algorithm if β is chosen to be N, the total number of triangles. When β is chosen to be a value between 2 and N-l, the behavior of our algorithm is also between these two traditional methods: the sort speed drops but the in-memory tree size decreases with increasing β.

Id. at 11 [0035]).

Appellants' Figure 10 is reproduced below.

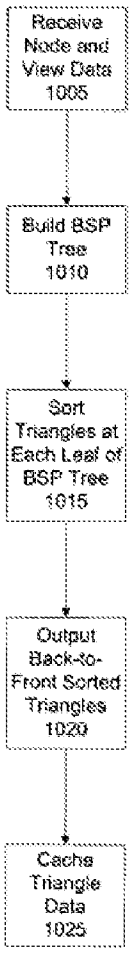


Figure 10

Figure 10 is described as a flowchart of a process in accordance with the preferred embodiment (Specification 7:1-2).

The process steps are described as follows at pages 32-33, paragraphs [0084-88]. First, the system receives geometry data describing a node and current view direction (step 1005). Next, the system constructs a BSP tree, where each triangle is associated with a leaf of the BSP tree (step 1010) in a

process discussed in detail *infra*. Then, the triangles are sorted into substantially back-to-front order by traversing the BSP tree and sorting the triangles at each leaf (step 1015). Next, the sorted triangles are output to the graphics API (step 1020). Finally, the sorted data for the current view, either in triangle form or in display list form, is cached (step 1025).

Appellants' Figure 3 is reproduced below.

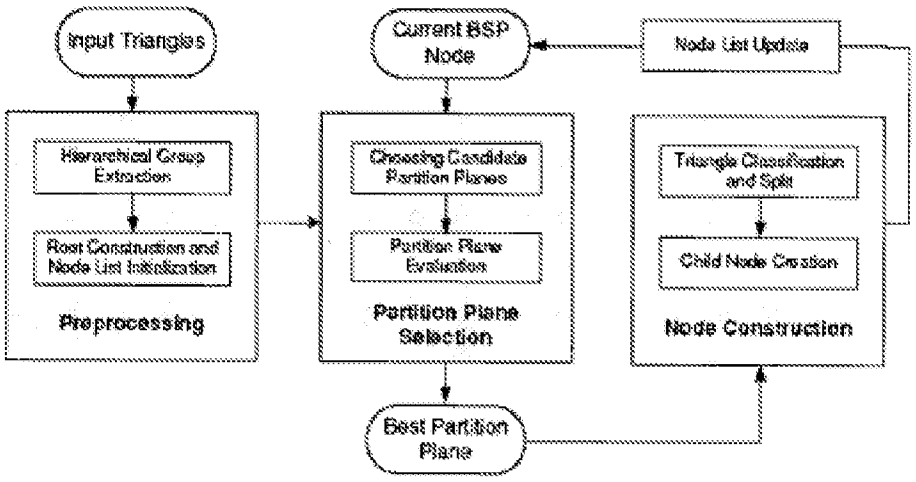


Figure 3

Figure 3 shows an overview of the algorithm for constructing a BSP tree in accordance with Appellants' invention (*id.* at 6 [0014]).

As indicated in this figure, the algorithm consists of three steps: preprocessing, partition plane selection, and node construction. The preprocessing step analyzes the normal (i.e., the normal to the triangle plane) and position information of the triangles (*id.* at 16:11-14) in order to organize the input triangles into coherent hierarchical groups conveying

certain feature or orientation pattern information that can be used to quickly identify candidate partition planes of high quality. *Id.* at 19:24-20:2.

Figure 5 is reproduced below.

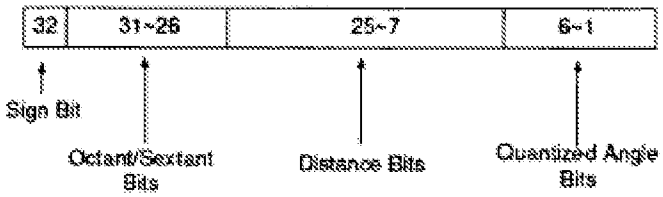


Figure 5

Figure 5 depicts an exemplary layout of a 32-bit plane index (*id.* at 6 [0016]).

Each triangle is analyzed and encoded as such a 32-bit integer plane index, which is composed of an octant index with value between 0 to 7, a sextant index with value between 0 to 5, and two angle values (*id.* at 20:15-21).

Figure 6 is reproduced below.

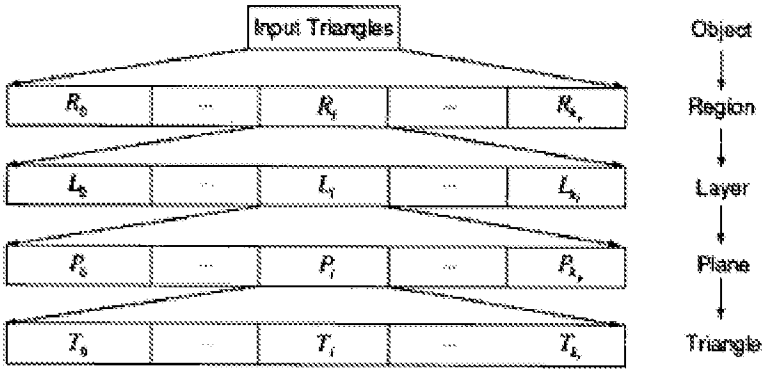


Figure 6

Figure 6 depicts an example of hierarchical groups extracted from the plane index data (*id.* at 6 [0017]).

The extraction process that produced the hierarchical groups depicted in Figure 6 is described as follows:

> The extraction process starts with sorting all the triangles with respect to their plane index (in descending order, without loss of generality) into a seed list S. The region groups, each of which contains a contiguous list of triangles in S with the same octant and sextant values[,] are then extracted from list S. From the definition of octant and sextant values, at most 48 regions exist and all the triangles in the same region have roughly the same orientation. As can be seen from Figure 5, within each region the triangles are sorted by their distance. The triangles having the same distance form a layer, and the layers in the same region are sorted by their distance in descending order. Within each layer, the triangles with the same quantized angle values, and therefore nearly identical orientation, are grouped into a plane. Each plane consists of one or more triangle elements.

Id. at 21 [0058].

By analyzing the normal and position information of the triangles in the foregoing manner, the preprocessing step extracts shape knowledge that is used to more effectively choose candidate partition planes and create a root node containing all of the input triangles (*id.* at 16:11-16).

The basic strategy for partition plane selection is "hypothesis and test," which involves comparing a selected set of candidate planes and choosing the one with the highest quality (*id.* at 21:25-28). According to the preferred embodiments, the extracted shape knowledge is exploited to

quickly select candidate planes having the best chance to be a good partition plane as evaluated by the criterion set forth in Equation (1) (*id.* at 21:28-22:2). That equation is as follows:

$$w = n_o - K_c n_c - \text{abs}(n_F - n_B) \qquad (1)$$

(*id.* at 19:8), where w is the partition plane criterion for a candidate partition plane P, n_o is the number of triangles on P, n_c is the number of triangles crossing P, n_F is the number of triangles on the positive side of P, n_B is the number of triangles on the negative side of P, K_c is a constant value that indicates the significance of triangle split, and symbol "abs" stands for the absolute value operation (*id.* at 17 [0048]; *id.* at 19 [0053]). The candidate partition plane that has the maximum w value should have the best chance of resulting in a shorter tree construction time and a better BSP tree (*id.* at 17:5-8).

Six candidate planes are selected for evaluation (*id.* at 22-23 [0060-61]). After the best partition plane has been chosen for the current BSP node, a front child or a back child or both are then created with necessary parent-child links (*id.* at 24-25 [0066]). All the triangles that are on the positive side of the partition plane are put on the front child, while all the triangles that are on its negative side are put on the back child (*id.*). Triangles that are on the partition plane will stay on the current node, and those triangles crossing the partition plane will be split into two or three triangles (*id.*). Each of the split triangles is either added to the front child or back child (*id.*).

The foregoing tree construction procedure is depicted in Figure 11, reproduced below.

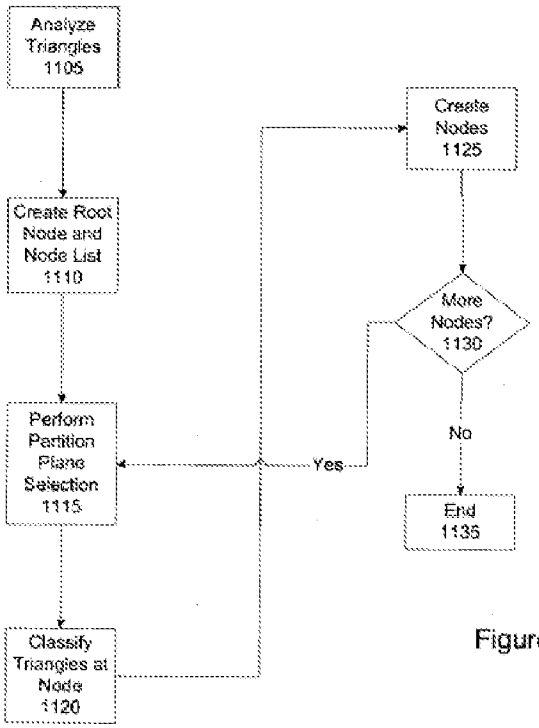


Figure 11

Figure 11 is described as depicting a flowchart of a process in accordance with a preferred embodiment (*id.* at 7 [0022]).

As shown in this figure, which is described at page 33, paragraphs [0089-91], in the preprocessing stage (step 1105) the system first analyzes the shape knowledge of each triangle, including the normal and position information of each triangle. Next, the system creates a root node and node list (step 1110). Then for each node in the tree, the system performs a partition plane selection (step 1115). Then, each triangle of the node is classified against the partition plane (step 1120), and according to this

classification, child nodes are created (step 1125). If there are remaining nodes (step 1130), the process repeats (returning to step 1115). If not, the process ends (step 1135), and the next step in the overall process can be performed.

The visibility order between the triangles organized in a BSP tree constructed in the above manner is computed by a back-to-front traversal of the BSP tree with respect to the eye point and a depth-sort procedure at each leaf node when it is being traversed (*id.* at 25:13-17).

B. The claims

Thee independent claims are claims 1, 8, 11, 18, 21, and 28, of which claim 1 reads:

> 1. A method for graphics processing, comprising:
>
> receiving node and view data for a graphic object;
>
> building a binary-space-partition tree corresponding to the graphic object, the binary-space-partition tree having up to a predetermined number of at least one shape associated with each leaf;
>
> sorting shapes at each leaf of the binary-space-partition tree; and
>
> outputting the sorted shapes.

Br., Claims App. 1.

The "up to a predetermined number of" language in claim 1 and the other independent claims was added by the May 26, 2006, Amendment and Response to Office Action. Before that amendment, claims 1, 11, and 21

recited "a binary-space-partition having at least one shape associated with each leaf," while claims 8, 18, and 28 recited "a binary-space-partition, each node associated with at least one shape."

C. The references and rejections

The references relied on by the Examiner are:

Brokenshire et al. (Brokenshire)	US 6,624,810 B2	Sep. 23, 2003
Vlasic et al. (Vlasic)	US 2004/0114794	Jun. 17, 2004
Korobkin	US 6,912,293 B1	Jun. 28, 2005

Claims 1, 2, 4, 5, 8-12, 14, 15, 18-22, 24, 25, and 28-30 stand rejected under 35 U.S.C. § 103(a) for obviousness over Korobkin in view of Brokenshire.

Claims 3, 6, 7, 13, 16, 17, 23, 26, and 27 stand rejected under § 103(a) for obviousness over Korobkin in view of Brokenshire and Vlasic.

THE ISSUE

The issue is whether Appellants have shown reversible error by the Examiner in maintaining the rejections. *See In re Kahn*, 441 F.3d 977, 985-86 (Fed. Cir. 2006) ("On appeal to the Board, an applicant can overcome a rejection by showing insufficient evidence of *prima facie* obviousness or by rebutting the *prima facie* case with evidence of secondary indicia of

nonobviousness.") (quoting *In re Rouffet*, 149 F.3d 1350, 1355 (Fed. Cir. 1998)).

ANALYSIS

A. The scope and meaning of the claims

The Examiner and Appellants have not expressed any disagreement about claim interpretation or explained how they are construing the various claim terms.

Independent claims 1, 11, and 21 employ the terms "binary-space-partitioning tree," "leaf," and "up to a predetermined number." These terms are not defined in the Specification and therefore must be given their broadest reasonable interpretations consistent with the disclosure. *In re Morris*, 127 F.3d 1048, 1054 (Fed. Cir. 1997). We understand "leaf" as used in the claims to mean a terminal node and therefore to exclude the root node and intermediate nodes. Furthermore, because these claims call for *sorting* shapes at each leaf and then *outputting* the sorted shapes, it is clear that the recited "binary-space-partitioning tree" refers to a completed BSP tree rather than a BSP tree that is still under construction. "Predetermined number" means a number that is determined prior to construction of the BSP tree. Finally, because a leaf cannot have zero associated shapes and in order to give meaning to "up to," we understand the phrase "up to a predetermined number" to mean "up to a predetermined number greater than one." As a result, the phrase "the binary-space-partition tree having up to a

predetermined number of at least one shape associated with each leaf" refers to a BSP tree in which each leaf has associated therewith up to a predetermined number greater than one of at least one shape (e.g., a triangle), with at least one leaf having associated therewith *the* predetermined number greater than one of at least one shape. The Specification explains that the phrases "associated with" and "associated therewith" and derivatives thereof "may mean to include, be included within, interconnect with, contain, be contained within, connect to or with, couple to or with, be communicable with, cooperate with, interleave, juxtapose, be proximate to, be bound to or with, have, have a property of, or the like" (Specification 4:27 to 5:4).

Independent claims 8, 18, and 28 employ the term "node" rather than "leaf." These claims specify that "each node [is] associated with up to a predetermined number of at least one shape." We are construing "node" as used in the claims to be broad enough to read on a root node, a leaf node, or an intermediate node. Furthermore, because a node cannot have zero associated shapes[4] and in order to give meaning to "up to," we understand the phrase "up to a predetermined number" to mean "up to a predetermined number greater than one." As a result, we understand the phrase "each node [is] associated with up to a predetermined number of at least one shape" to

[4] The root node is associated with all of the shapes in the tree. An intermediate node is associated with all of the shapes in its corresponding subtree. A leaf node has at least one associated shape.

mean each node has associated therewith up to a predetermined number greater than one of at least one shape (e.g., a triangle), with at least one node having associated therewith *the* predetermined number greater than one of at least one shape.

B. The merits of the rejections

Korobkin discloses a graphics system for visualizing the placement of one or more physical objects in a physical scene (Korobkin, col. 1, ll. 52-54).

Korobkin explains that

> [o]ne use of the graphics system is for electronic commerce. For example, a consumer captures a digital image of a room in the user's home and provides it to the graphics system. A merchant captures digital images of products it has for sale. Thus, the graphics system can combine the images to show the consumer an image of what the product might look like within the room where the user might place the products, if purchased.
>
> If the user cannot capture all of the room in one image, the user can capture multiple images with some overlap and the graphics system will match them up to form a "mosaic" image that can then be mapped to a geometric model.

Id. at col. 1, l. 64 to col. 2, l. 8.

A Visual Reconstruction Engine 16 (Fig. 2) automatically applies 2D input imagery onto 3D geometric models using recovered 3D camera models (*id.* at col. 13, ll. 43-45). For processing of 3D texture-mapped geometric object and scenes from one or more 2D images, visual reconstruction processor 16 produces a graphical database that includes camera maps,

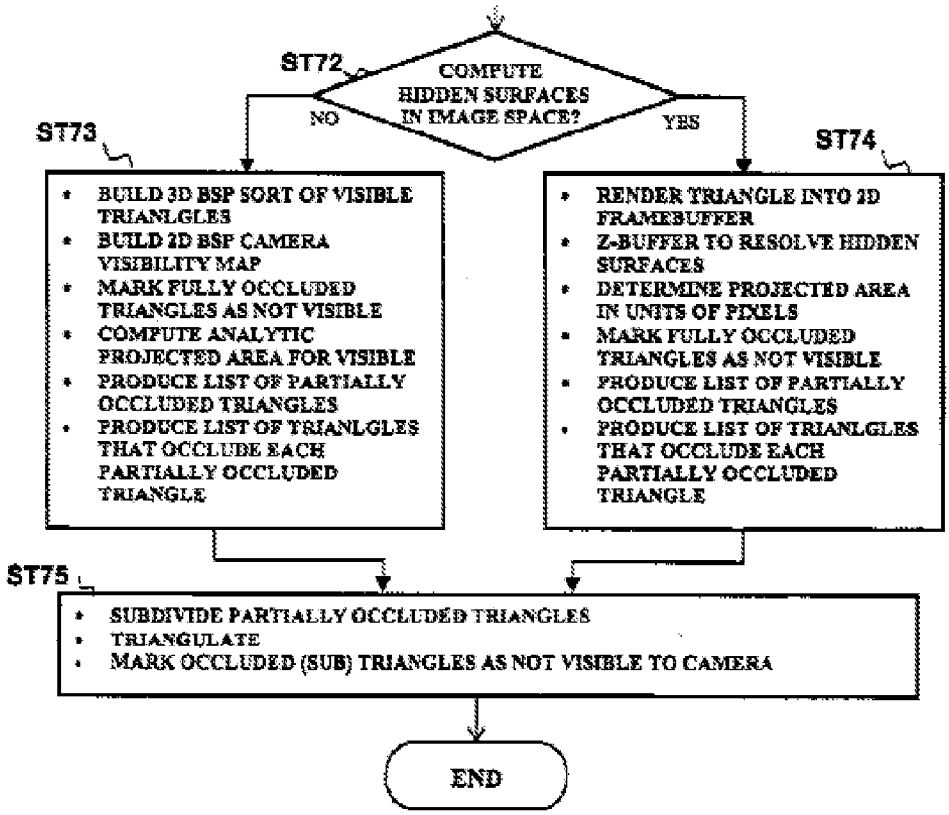
texture maps and coordinates, camera models, and 3D object and scene geometry (*id.*, col. 14, ll. 6-10). A camera map is an assignment of cameras to scene geometry (*id.*, col. 14, ll. 10-11). For each triangular facet of model geometry seen by at least one camera, a camera map produced by visual reconstruction processor 16 designates a single "best" camera or group of cameras whose imagery is available and appropriate to texture map the triangle (*id.*, col. 14, ll. 11-14).

Visibility processing is required to determine, for each camera, which facets of scene geometry (in whole or in part) are visible from the viewpoint of the camera (*id.*, col. 14, ll. 16-19). The disclosed visual reconstruction technique employs a hybrid visibility processing approach, invoking both object-space and image-space visibility computation (*id.*, col. 14, ll. 35-37). The first and second stages of the visual reconstruction process are depicted in flowchart form in Figures 15 and 16, respectively (*id.*, col. 14, ll. 57-61; *id.*, col. 16, ll. 35-37).

The first stage of the visual reconstruction process is a determination of the visible triangles relative to each scene camera (*id.*, col. 14, ll. 57-60). This first stage is carried out for each camera (*id.*, col. 14, ll. 60-61). In step ST70 (Fig. 15), all back-facing triangles are marked as not visible (*id.*, col. 14, ll. 62-67). In step ST71, the remaining triangles are compared to the frustum of the camera (*id.*, col. 15, ll. 1-12). Triangles that intersect the camera frustum are divided into sub-triangles (*id.*, col. 15, ll. 7-9). Triangles

and sub-triangles that are outside the camera frustum are marked as not visible (*id.*, col. 15, ll. 11-13).

Figure 15 is reproduced in part below.



The portion of Figure 15 reproduced above includes the blocks that depict steps ST72-75.

At step ST72, the system or the user selects either image-space computation or object-space computation for resolution of hidden surfaces (*id.*, col. 15, ll. 21-23). If an object-space computation is selected, process flow proceeds to step ST73, which employs an object-space algorithm (*id.*, col. 15, ll. 26-27); otherwise, process flow proceeds to step ST74 (*id.*,

col. 15, ll. 23-25), which employs an image-space algorithm (*id.*, col. 16, ll. 4-5). Step ST73 employs "an object-space algorithm [that] computes analytic visibility in two stages. The first stage is a sort of the triangle database in 3-space. The second stage is a determination of the visible triangle fragments in a camera imaging screen." *Id.*, col. 15, ll. 26-30. The first of these two stages takes the form of a "3D Scene Visibility Tree," which appears to be a conventional three-dimensional BSP:

> 3D Scene Visibility Tree
>
> The system utilizes a 3D binary space partition (BSP) tree to accomplish a global visibility sort of the input database. The BSP tree sort recursively subdivides the object space and geometry with hyper-planes defined by the surface facets of the input[.] From any given arbitrary viewpoint, a subsequent traversal of the tree will deliver triangles in a spatially correct "back-to-front" or "front-to-back" ordering.

Id., col. 15, ll. 31-38. In the second stage, "[t]he scene visibility tree is traversed in a front-to-back order. Each triangle encountered is ray-projected into the camera imaging plane and inserted in a 2D BSP referred to as [a] 2D camera visibility map" (*id.*, col. 15, ll. 39-42),[5] which Korobkin describes as follows:

> 2D Camera Visibility Map
>
> A camera visibility map depicts the visible geometry of the scene, as viewed by a camera and projected into its imaging plane (screen). In the event of inter-object occlusions, the

[5] Projection of a line element onto a camera plane is depicted in Figure 10(a), discussed at column 12, lines 16-37.

> visibility map resolves hidden surfaces, depicting resulting visible geometry fragments. A camera visibility map is constructed for each camera in the scene. The object-space camera visibility map is encoded as a 2D BSP tree. In two-dimensions, a BSP tree partitions a camera screen. Edges of the input geometry projected onto the imaging plane define the lines partitioning the space.
>
> The screen of the camera is partitioned into regions that are occupied with projected geometry (G-regions), and those unoccupied (U-regions). When a polygon (triangle) is inserted, it is and clipped [sic] to visited U-regions. In the process, the clipped visible region of the polygon overlapping the U-region becomes a G-region. That is, it is removed from the visible region of the screen.
>
> For a triangle, 3 edges become 3 line partitions. Each line recursively subdivides the camera screen into two new sub-planes.
>
> G-regions and U-regions appear as leaf nodes in the tree. G-regions are tagged to identify their specific origin. Insertion of geometry stops when there are no U-region nodes (screen is full). Intersections of input geometry and previously inserted geometry (G-regions) [are] detected. This determines, for a given triangle, which, if any other triangles obscure it. Fully occluded geometry is identified and tagged.

Korobkin, col. 15, l. 43 to col. 16, l. 3.

The second stage of the visual reconstruction process, illustrated in Figure 16, is an assignment of cameras to triangles for texture mapping (*id.* at col. 16, ll. 35-37). For each triangle and for each camera which "sees" the triangle, the angle between triangle normal and camera view direction is calculated (*id.* at col. 16, ll. 38-40). In one embodiment, the camera

presenting minimum view angle and maximum projective area is selected to map onto a given triangle (*id.* at col. 16, ll. 40-42).

(1) Claims 1-7, 11-17, and 21-27

The rejection of claim 1 is based on Korobkin's 3D scene visibility tree and also on Korobkin's 2D camera visibility map, which as noted above also takes the form of a BSP tree (*id.*, col. 15, ll. 50-51). Specifically, the Examiner's reliance on Korobkin's 3D scene visibility map is indicated by citations to Korobkin's discussions of that BSP tree at column 15, lines 33-36 (Answer 3) and column 15, lines 32-38 (*id.* at 4) as well as the Examiner's reliance on "a global visibility sort" (*id.*), which is the function of that BSP tree (Korobkin, col. 15, ll. 32-33). The Examiner's reliance on the 2D camera visibility map is indicated by the discussion of G-regions and U-regions (Answer 3), which are leaves in the BSP tree that represents the 2D camera visibility map. *See* Korobkin, col. 15, l. 64 ("G-regions and U-regions appear as leaf nodes in the tree").

The rejection is based on alternative rationales, of which the first is based on Korobkin's 2D camera visibility map:

> Korobkin implicitly[6] discloses that G-regions and U-regions as leaf nodes in the tree. Insertion of geometry stops; it means the partition having up to a predetermined number and stops the partition; as one of ordinary skill in the art, the partition of the binary space tree either in object-space or image-space[7]

---

[6] As noted above, this disclosure is explicit rather than implicit.

[7] This mention of image-space appears to be incorrect. Korobkin's

(Continued on next page.)

> should be continued until a desired or predetermined number of levels of subspace or sub-object (leaf or child) has been created and stopped the partition for a valid scene graph database; . . . .

Answer 8. This rationale is unconvincing for several reasons. In the first place, Korobkin does not disclose that partitioning of the 2D camera visibility map continues until a predetermined number of levels of subspaces or sub-objects has been created. Rather, Korobkin discloses that partitioning continues until there are no more U-region nodes (i.e., the screen is full), i.e., all areas of the screen are encoded as G-region nodes. The Examiner has not explained why, and it is not otherwise apparent that, this final number of G-region nodes will be "predetermined," i.e., determined prior to encoding the screen as BSP tree. Furthermore, even assuming for the sake of argument that the final number of G-region nodes is predetermined, the Examiner has not explained why, and it is not otherwise apparent that, each leaf of the final G-region node BSP tree will inherently have up to a predetermined number greater than one of triangles associated therewith, with at least one leaf having *the* predetermined number greater than one of triangles associated therewith.

The Examiner alternatively relies on Brokenshire for an explicit teaching of dividing a space into a predetermined number of subspaces when forming a BSP tree (Answer 4). Brokenshire discloses a method and an

---

BSP trees are used in object-space computations (step ST73) rather than in image-space computations (step ST74).

apparatus for improving the accuracy of the bounding volumes for BSP trees (Brokenshire, col. 1, ll. 21-23).

Figures 3A and 3B of Brokenshire are reproduced below.

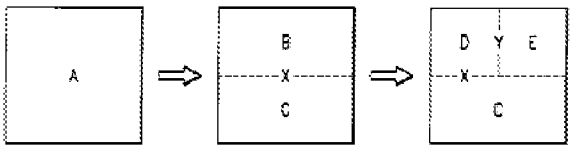


FIG. 3A

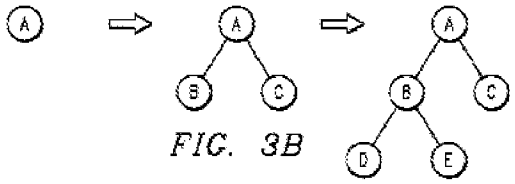


FIG. 3B

Figure 3A shows a two-dimensional object (plane A) subdivided by "hyperplanes" to form subspaces, while Figure 3B shows development of the resulting BSP tree (*id.*, col. 2, ll. 20-23).

Figures 6A and 6B of Brokenshire are reproduced below.

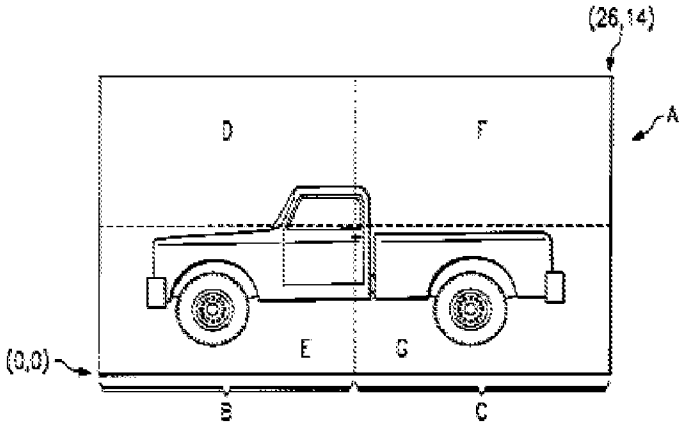


FIG. 6A

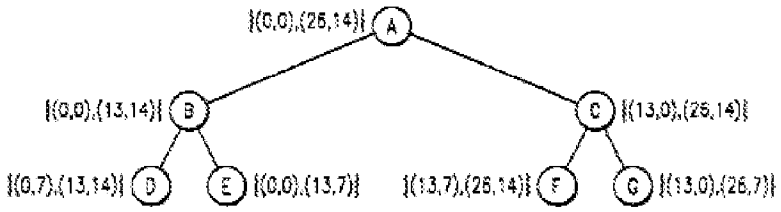


FIG. 6B

Figure 6A shows a two-dimensional plane A containing an image of a pickup truck (*id.*, col. 5, ll. 51-52). while Figure 6B shows the BSP tree that is constructed in accordance with the subspaces depicted in Figure 6A (col. 6, ll. 1-2).

Plane A in Figure 6A is initially subdivided into subplanes B and C, after which subplane B is subdivided into subplanes D and E, and then subplane C is subdivided into subplanes F and G (*id.*, col. 5, ll. 53-56). These polytopes are formed in a computationally efficient manner by bisecting each plane in the opposite direction of the previous bisection (*id.*, col. 5, ll. 57-59). This approach is simple but does not take into

consideration the image being displayed (*id.*, col. 5, ll. 57-60). The result of forming subplanes B-G in the manner shown in Figure 6A (*id.*, col. 6, ll. 12-14) is that subplane D will have a lot of "dead space," which means the subplane has a large area that does not overlap the image of the pickup truck (*id.*, col. 6, ll. 14-16). The result is a computational disadvantage that is addressed below.

Figures 7A and 7B are reproduced below.

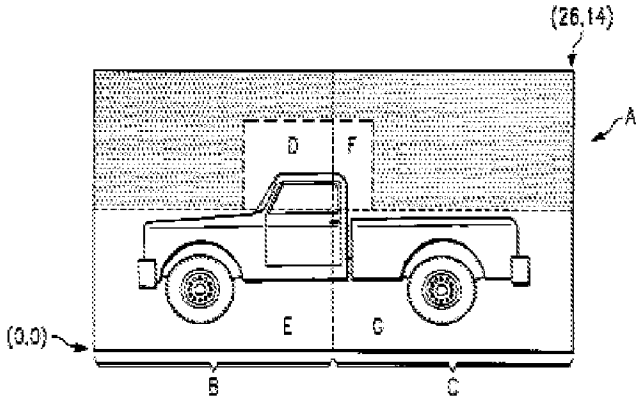


FIG. 7A

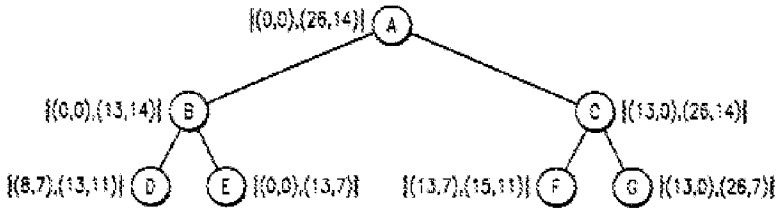


FIG. 7B

Figure 7A shows the same two-dimensional plane A containing an image of a pickup truck as in FIG. 6A (*id.*, col. 6, ll. 11-12), while Figure 7B

shows the BSP tree constructed in accordance with the subspaces of Figure 7A (*id.*, col. 6, ll. 21-22).

As shown in Figure 7A, subplane D can be reduced in size relative to its size in Figure 6A and still cover the image of the pickup truck, while, in a similar manner, subplane F can be reduced in size and still cover the image of the pickup truck (*id.*, col. 6, ll. 16-20). Reducing the sizes of subplanes D and F in this manner can result in a reduction in the time required to render and display shifted versions of the truck image:

> [W]ith the boundaries as determined using the present invention, if the image of the pickup truck were moved to the right to such an extent that the entire reduced subspace D as shown in FIG. 7A is removed from the viewable area of the display, the rendering program is able to determine that it does not need to render the contents of reduced subspace D since that portion of the image will not be displayed. Therefore, the program needs only to render the image in subspace E, a task which requires much less bus traffic between memory or an I/O device and the processor and which requires much less computations. Thus, the rendering and displaying of an image may be performed more quickly by reducing the bounding values of a subspace in accordance with the present invention.

Brokenshire, col. 7, ll. 28-41.

Brokenshire's Figure 8 is reproduced below.

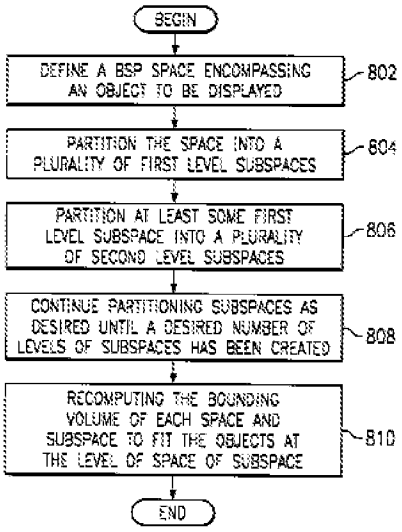


FIG. 8

Figure 8 is a flowchart illustrating an exemplary process and program flow for determining which subspaces should have their bounds tightened (*id.*, col. 6, ll. 32-36). First, a space, or root node, within a binary space partitioning tree is created for an object for which an image is to be rendered (step 802) (*id.*, col. 6, ll. 39-42). The space should include the entire bounds of the object that is to be rendered and displayed (*id.*, col. 6, ll. 42-43). Next, the space is partitioned into a plurality of first level subspaces to create child nodes or child subspaces (step 804) (*id.*, col. 6, ll. 43-45). Each first level subspace may then be partitioned into a plurality of second level subspaces (step 806) (*id.*, col. 6, ll. 45-47). The rendering program continues partitioning each subspace, if desired, such that at each level, each subspace is partitioned into further subspaces until a desired number of levels of subspaces has been created (step 808) (*id.*, col. 6, ll. 52-5). Once

the space has been subdivided into a predetermined number of level of subspaces, the bounding volume of each space or subspace is recomputed such that the bounding volume just contains the object or objects that fit into that level of the BSP tree (step 810) (*id.*, col. 7, ll. 3-7).

The Examiner, citing the foregoing teachings of Brokenshire, concluded that it would have been obvious, when forming Korobkin's 2D camera visibility map, to divide the camera screen into a predetermined number of levels of subspaces (Answer 4). We agree with Appellants that when the teachings of Korobkin and Brokenshire are combined in this manner, the resulting BSP tree will not have any leaves that represent multiple triangles (Reply Br. 9) and thus will not satisfy the requirement of claim 1 for leaves representing up to a predetermined number greater than one of triangles, as required by claim 1.

In an additional rationale for the rejection, the Examiner, relying on the global visibility sort provided by Korobkin's 3D scene visibility tree, concluded that

> it would have been obvious to one . . . to incorporate the method of binary-space partitioning tree of Brokenshire into the BSP tree of Korobkin to accomplish a global visibility sort of the input database (Korobkin; col. 15, lines 22-23), because once the space has been subdivided into a predetermined number of level[s] of subspaces, the bounding volume of each space or subspace is recomputed such that the bounding volume just contains the object or objects that fit into that level of BSP tree.

Answer 4-5. The Examiner has not explained why, and it is not otherwise apparent that, forming Korobkin's 3D scene visibility tree in this manner will yield any leaves that represent multiple triangles.

For the foregoing reasons, the rejection for obviousness based on Korobkin in view of Brokenshire is reversed with respect to claim 1, with respect to claims 11 and 21, which are similar to claim 1, and also with respect to dependent claims 2, 4, 5, 12, 14, 15, 22, 24, and 25.

The Examiner has not explained why the foregoing deficiency in the combined teachings of Korobkin and Brokenshire is remedied by Vlasic, which is additionally relied on in the rejection of dependent claims 3, 6, 7, 13, 16, 17, 23, 26, and 27. The rejection of those claims is therefore also reversed.

(2) Claims 8-10, 18-20, and 28-30

Claim 8 reads as follows:

> 8. A method for graphics processing, comprising:
>
> analyzing shapes in a graphic object;
>
> creating a root node and a list of additional nodes for a binary-space-partition tree, each node associated with up to a predetermined number of at least one shape;
>
> performing a partition plane selection for each additional node,
>
> classifying the shapes at the additional node according to the partition plane selection; and
>
> creating child nodes according to the shape classification.

Br., Claims App. 2.

Claim 8 is broader than claim 1 in two significant respects. First, the "associated with up to a predetermined number of at least one shape" in claim 8 applies to nodes in general rather being than being limited to leaves. Second, unlike claim 1, claim 8 does not call for sorting shapes at a node, let alone sorting shapes at a leaf node. We also note that the step of "analyzing shapes" is not limited in time with respect to performance of the other recited steps and can therefore occur prior to or concurrently with those other steps.

The rejection of claim 8 is based on combining the teachings of Korobkin and Brokenshire in the same manner as in the rejection of claim 1. *See* Answer 9 ("[Regarding] Claim 8, the discussion of claim 1 above is incorporated hereto.").

Appellants argue:

> This claim requires, among other limitations, classifying the shapes at the additional node according to the partition plane selection. The Examiner simply refers to Korobkins' [sic] teaching regarding the G-regions and U-regions in the camera view, where each region that is occupied with projected geometry is designated a G-region, and unoccupied regions are U-regions. Even if this were regarded as a "partition plane selection" for the U-region and G-region nodes, no shapes at the nodes are classified according to this partition plane selection, as claimed, and no child nodes are created according to any such shape classification, as claimed.

Reply Br. 10-11. This argument is unconvincing for the following reasons. Korobkin explains that "[e]dges of the input geometry projected onto the imaging plane define the lines partitioning the space" (Korobkin, col. 15, ll.

52-53). Appellants have not explained why this does not amount to performing a "partition plane selection." Korobkin also explains: "When a polygon (triangle) is inserted, it is and clipped [sic] to visited U-regions. In the process, the clipped visible region of the polygon overlapping the U-region becomes a G-region. *Id.* at col. 15, ll. 56-59. Appellants have not explained why the designation of a polygon region as part of a U-region or a G-region does not constitute "classifying" a shape according to the partition planes. Korobkin further explains that "G-regions and U-regions appear as leaf nodes in the tree" (*id.*, col. 15, l. 64). Appellants have not explained why generating a BSP tree containing such leaf nodes would not inherently involve the creation of child nodes.

Nor have Appellants addressed the Examiner's modification of Korobkin's 3D scene visibility tree (i.e., global visibility sort) in view of Brokenshire so as to subdivide the space into a predetermined number of levels of subspaces, coupled with recomputing the bounding volume of each space or subspace (Answer 4-5). Thus, Appellants have not explained, for example, why the analyzing step of claim 8 cannot be read on: (a) step ST70 in Korobkin's Figure 15, during which all back-facing triangles are marked as not visible (*id.*, col. 14, ll. 62-67); (b) step ST71, in which all triangles and sub-triangles that are outside the camera frustum are marked as not visible (*id.*, col. 15, ll. 11-13); or (c) the recomputing step added by Brokenshire. Nor have Appellants explained why each node in the resulting 3D scene visibility tree does not represent (i.e., is not associated with) up to

a predetermined number of triangles, with the predetermined number being the total number of triangles. The root node corresponds to all of the triangles, each leaf node corresponds to a single triangle, and each intermediate node corresponds to the number of triangles in its subtree. Appellants also have not explained why the recited "classifying" step does not read on classifying each shape as being either behind or in front of the hyperplane for the associated node.

Nor have Appellants explained why claim 8 cannot be read on the BSP tree-forming procedure depicted in Brokenshire's Figures 7A and 7B in a similar manner. The recited "analyzing" step appears to read on recomputing the sizes of the subplanes to eliminate "dead spaces" (Brokenshire, Fig. 8, block 810). The step of "creating a root node and a list of additional nodes for a binary-space-partition tree, each node associated with up to a predetermined number of at least one shape," appears to read on creating a root node A and listing nodes B-G as additional nodes. Each of nodes A-G can be fairly characterized as "associated with up to a predetermined number of at least one shape," because each node is associated with up to four different shapes (i.e., subplanes). Specifically, node A is associated with four subplanes (D-G); node B is associated with two subplanes (D and E); node C is associated with two subplanes (F and G); and nodes D-G are associated with subplanes D-G, respectively. The step of "performing a partition plane selection for each additional node" appears to read on any manner of selecting the positions of the hyperplanes

that are used to create the subplanes, while the step of "classifying the shapes at the additional node according to the partition plane selection" appears to read on classifying each subplane as being either left or right (or either above or below) its respective hyperplane. Finally, the step of "creating child nodes according to the shape classification" appears to read on creating nodes B-G based on the shape classifications.

For the foregoing reasons, we are affirming the rejection of independent claim 8 for obviousness over Korobkin in view of Brokenshire and also the rejection of independent claims 18 and 28, which are rejected over the same prior art and as to which Appellants repeat their claim 8 argument. The rejection of unargued dependent claims 9, 10, 19, 20, 29, and 30 based on the same prior art is also affirmed. *In re Young*, 927 F.2d 588, 590 (Fed. Cir. 1991); 37 C.F.R. § 41.37(c)(1)(vii) (2007).

DECISION

The rejection of claims 1, 2, 4, 5, 11, 12, 14, 15, 21, 22, 24, and 25 under 35 U.S.C. § 103(a) for obviousness over Korobkin in view of Brokenshire is reversed. The § 103(a) rejection of claims 8-10, 18-20, and 28-30 for obviousness over those references is affirmed.

The § 103(a) rejection of claims 3, 6, 7, 13, 16, 17, 23, 26, and 27 for obviousness over Korobkin in view of Brokenshire and Vlasic is reversed.

The decision of the Examiner is accordingly affirmed-in-part.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a). *See* 37 C.F.R. §§ 41.50(f) and 41.52(b).

AFFIRMED-IN-PART

rvb

DOCKET CLERK
PO BOX 800889
DALLAS, TX 75380